

JOBSHEET 12

SEARCHING DAN SORTING

12.1 Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Menjelaskan mengenai algoritma Searching
2. Membuat dan mendeklarasikan struktur algoritma Searching
3. Menerapkan dan mengimplementasikan algoritma Searching
4. Mahasiswa mampu memahami algoritma sorting pada jenis Bubble Sort.
5. Mahasiswa mampu membuat dan mendeklarasikan struktur algoritma Bubble sort
6. Mahasiswa mampu menerapkan dan mengimplementasikan algoritma Bubble sort

12.2 Unordered Sequential Search pada Array

Waktu percobaan: 20 menit

12.2.1 Langkah-langkah Percobaan

1. Buat class dengan nama **UnorderedSearch**. Pada class tersebut, buat fungsi untuk pencarian dengan parameter input adalah array of integer dan sebuah bilangan integer yang akan dicari apakah ada atau tidak pada array. Lalu buat fungsi utama (main) untuk melakukan pencarian sebuah bilangan bulat dengan memanggil fungsi search tersebut, seperti pada gambar berikut:

```
public class UnorderedSearch {  
  
    static int search(int[] arr, int num){  
  
        for (int i = 0; i < arr.length; i++){  
            if (arr[i] == num)  
                return i;  
        }  
        return -1;  
    }  
  
    public static void main(String[] args) {  
        int[] arrInt = {10, 5, 20, 2, 11, 8, 4, 15};  
        int cari = 8;  
        int hasil = search(arr:arrInt, num:cari);  
        if (hasil == -1)  
            System.out.println("Elemen "+cari+" tidak ditemukan");  
        else  
            System.out.println("Elemen "+cari+" ditemukan pd posisi ke-"+hasil);  
    }  
}
```

2. Jalankan (run) class **UnorderedSearch** dan amati hasilnya.
3. Ganti variabel cari dengan bilangan 7, lalu run program. Amati hasilnya.

12.2.2 Verifikasi Hasil Percobaan

Cocokkan hasil run kode program anda jika cari = 8, dengan gambar berikut ini.

```
Elemen 8 ditemukan pd posisi ke-5
-----
BUILD SUCCESS
-----
```

Jika cari = 7, maka hasil run sebagai berikut:

```
Elemen 7 tidak ditemukan
-----
BUILD SUCCESS
-----
```

12.2.3 Pertanyaan

1. Pada percobaan di atas, jika cari = 8 dan cari = 7, maka masing-masing berapa kali banyak perbandingan yang dilakukan?
2. Modifikasi kode program di atas dengan menerima nilai variabel **cari** secara dinamis (berdasarkan nilai yang dimasukkan pengguna)

12.3 Unordered Sequential Search pada Array of Object

Waktu percobaan: 40 menit

12.3.1 Langkah-langkah Percobaan

1. Buat class baru dengan nama **Mahasiswa**. Tambahkan atribut kode, nim, namaMahasiswa dan ipk, serta berikan konstruktor berparameter seperti gambar berikut ini.

```
public class Mahasiswa {

    public String nim;
    public String namaMahasiswa;
    public double ipk;

    public Mahasiswa(String id, String name, double gpa){
        nim=id;
        namaMahasiswa=name;
        ipk=gpa;
    }

}
```

2. Buat class dengan nama **MahasiswaMain**. Buat class main, kemudian lakukan instansiasi objek array dan isikan datanya. Kemudian tambahkan kode program untuk melakukan pencarian menggunakan *Sequential Search*.

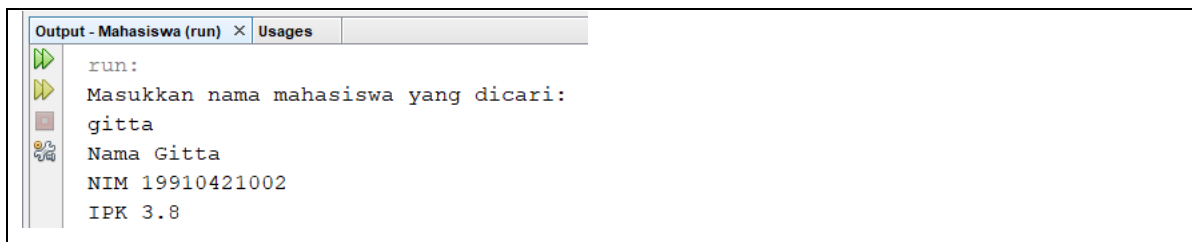
```
public class MahasiswaMain {
    public static void main(String[] args) {
        Mahasiswa[] mhs = new Mahasiswa[10];
        mhs[0] = new Mahasiswa("19970506001", "Alfatih", 3.9);
        mhs[1] = new Mahasiswa("19910421002", "Gitta", 3.8);
        mhs[2] = new Mahasiswa("19950322002", "Amanda", 3.75);
        mhs[3] = new Mahasiswa("19980129001", "Kevin", 3.5);
        mhs[4] = new Mahasiswa("19990208002", "Windy", 3.92);
        mhs[5] = new Mahasiswa("19970710001", "Belva", 3.6);
        mhs[6] = new Mahasiswa("19920602001", "James", 3.7);
        mhs[7] = new Mahasiswa("19901106002", "Aruna", 3.3);
        mhs[8] = new Mahasiswa("19981005002", "Sabrina", 3.55);
        mhs[9] = new Mahasiswa("19991201001", "Agus", 3.65);

        Scanner sc = new Scanner(System.in);
        System.out.println("Masukkan nama mahasiswa yang dicari: ");
        String cari = sc.nextLine();
        for (int i = 0; i < mhs.length; i++) {
            if (cari.equalsIgnoreCase(mhs[i].namaMahasiswa)) {
                System.out.println("Nama " + mhs[i].namaMahasiswa);
                System.out.println("NIM " + mhs[i].nim);
                System.out.println("IPK " + mhs[i].ipk);
            }
        }
    }
}
```

3. Jalankan (run) class **MahasiswaMain** dan amati hasilnya.

12.3.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.



```
Output - Mahasiswa (run) x Usages
run:
Masukkan nama mahasiswa yang dicari:
gitta
Nama Gitta
NIM 19910421002
IPK 3.8
```

12.3.3 Pertanyaan

1. Perhatikan class **MahasiswaMain**, jelaskan fungsi kode program berikut ini!


```
if (cari.equalsIgnoreCase(mhs[i].namaMahasiswa))
```
2. Lakukan modifikasi pada program tersebut, sehingga jika data yang dicari tidak ditemukan, maka akan menampilkan "Data tidak ditemukan"!
3. Lakukan modifikasi pada program tersebut agar pencarian dilakukan bukan berdasarkan nama mahasiswa, melainkan berdasarkan IPK!

12.4 Ordered Sequential Search pada Array of Object

Waktu percobaan: 30 menit

12.4.1 Langkah-langkah Percobaan

1. Buat class baru dengan nama **OrderedSearch**. Tambahkan atribut arr, serta berikan konstruktor berparameter. Tambahkan pula method Cari dan Tampilkan seperti gambar berikut ini.

```
public class OrderedSearch {
    public double[] arr;

    public OrderedSearch(double arrayNilai[]) {
        arr = new double[arrayNilai.length];
        for(int i = 0; i<arr.length; i++){
            arr[i]=arrayNilai[i];
        }
    }

    public int cari(double keyword) {
        int index = -1;
        for(int i = 0; i<arr.length; i++){
            if(keyword==arr[i]){
                index=i;
                break;
            } else{
                if(keyword<arr[i]){
                    break;
                }
            }
        }
        return index;
    }

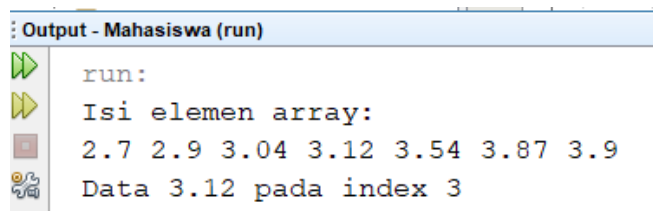
    public void tampilkan() {
        for(int i=0; i<arr.length;i++){
            System.out.print(arr[i]+" ");
        }
        System.out.println("");
    }
}
```

2. Buat class baru dengan nama **PencarianMain** tetap pada package yang sama dengan poin no 1. Buat class main, buat array dan isikan datanya seperti gambar berikut ini.

```
public class PencarianMain {
    public static void main(String[] args) {
        double[] data={2.7, 2.9, 3.04, 3.12, 3.54, 3.87, 3.9};
        OrderedSearch os= new OrderedSearch(data);
        System.out.println("Isi elemen array:");
        os.tampilkan();
        double key=3.12;
        int index= os.cari(key);
        if(index!=-1){
            System.out.println("Data "+ key+ "pada index "+ index);
        } else{
            System.out.println("Data "+ key+" tidak ditemukan");
        }
    }
}
```

3. Jalankan (run) class **PencarianMain** dan amati hasilnya.

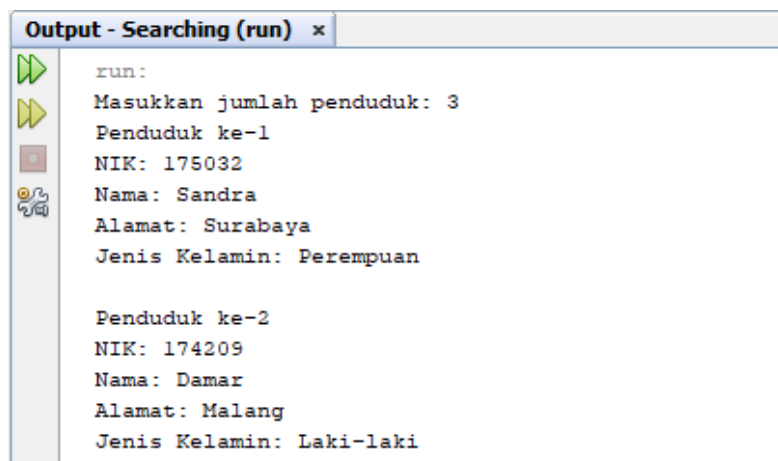
12.4.2 Verifikasi Hasil Percobaan



```
run:
Isi elemen array:
2.7 2.9 3.04 3.12 3.54 3.87 3.9
Data 3.12 pada index 3
```

12.4.3 Pertanyaan

1. Perhatikan class **PencarianMain**, jelaskan fungsi dari kode kode program berikut ini!
`OrderedSearch os= new OrderedSearch(data);`
2. Perhatikan class **Pencarian**, pada method **Cari** terdapat dua kali **break**. Jelaskan fungsi masing-masing break tersebut pada kode program tersebut!
3. Lakukan modifikasi pada program tersebut sehingga data array dapat bersifat dinamis (dapat diinputkan oleh pengguna)!



```
run:
Masukkan jumlah penduduk: 3
Penduduk ke-1
NIK: 175032
Nama: Sandra
Alamat: Surabaya
Jenis Kelamin: Perempuan

Penduduk ke-2
NIK: 174209
Nama: Damar
Alamat: Malang
Jenis Kelamin: Laki-laki
```

```
Penduduk ke-3
NIK: 154023
Nama: Rohman
Alamat: Surabaya
Jenis Kelamin: Laki-laki

--- DATA PENDUDUK ---
NO      NIK      NAMA      ALAMAT  JENIS KELAMIN
1       175032   Sandra   Surabaya Perempuan
2       174209   Damar    Malang   Laki-laki
3       154023   Rohman   Surabaya Laki-laki

--- CARI DATA PENDUDUK ---
Masukkan data yang dicari: 174209
      174209   Damar    Malang   Laki-laki
BUILD SUCCESSFUL (total time: 1 minute 15 seconds)
```

12.5 Sorting menggunakan Bubble Sort

Waktu percobaan: 30 menit

12.5.1 Langkah-langkah Percobaan

1. Buat class **Sort**, kemudian deklarasikan variabel berikut ini

```
//
public class Sort {
    public int [] data;
    public int jumData;
```

2. Buatlah konstruktor dengan parameter Data[] dan jumData

```
public Sort(int Data[], int jmlData){

}

}
```

3. Isi Konstruktor tersebut dengan kode berikut

```
public Sort(int Data[], int jmlData){
    jumData=jmlData;
    data=new int[jmlData];
    for (int i=0; i<jumData; i++){
        data[i]=Data[i];
    }
}
```

4. Buatlah method **bubbleSort** bertipe void
5. Deklarasikan isi method **bubbleSort** dengan menggunakan algoritma Bubble Sort

```
void bubbleSort(){
    int temp=0;
    for (int i=0; i<jumData-1; i++){
        for (int j=1; j<(jumData-i); j++){
            if (data[j-1]>data[j]){
                temp=data[j];
                data[j]=data[j-1];
                data[j-1]=temp;
            }
        }
    }
}
```

6. Buatlah method **tampilData** bertipe void

```
public void tampilData()
```

7. Deklarasikan isi dari method **tampilData**

```
public void tampilData(){
    for (int i=0;i<jumData;i++){
        System.out.print(data[i]+" ");
    }
    System.out.println();
}
}
```

8. Buat array pada class **Main** dengan nama data kemudian isi array tersebut

```
int a[]={15,10,7,22,5};
```

9. Buatlah objek baru dengan nama **urut** yang merupakan instansiasi dari class **Sort**, kemudian isi parameternya

```
Sort urut=new Sort(a, a.length);
```

10. Lakukan pemanggilan method **bubbleSort** dan **tampilData**

11. Jalankan program, dan amati hasilnya!

12.5.2 Verifikasi Hasil Percobaan

```
Data sebelum urut
15 10 7 22 5
Data sesudah urut Bubble sort (ASC)
5 7 10 15 22
```

12.5.3 Pertanyaan

1. Tunjukkan pada bagian mana proses pertukaran data terjadi dalam Bubble Sort!
2. Jelaskan fungsi penggunaan nested loop pada method **bubbleSort()**!